Computers, Materials & Continua CMC, vol.65, no.3, pp.2687-2705, 2020

Information Flow Security Models for Cloud Computing

Congdong Lv[1, *], Ji Zhang[2], Zhoubao Sun[1] and Gang Qian[1]

Abstract: Cloud computing provides services to users through Internet. This open mode not only facilitates the access by users, but also brings potential security risks. In cloud computing, the risk of data leakage exists between users and virtual machines. Whether direct or indirect data leakage, it can be regarded as illegal information flow. Methods, such as access control models can control the information flow, but not the covert information flow. Therefore, it needs to use the noninterference models to detect the existence of illegal information flow in cloud computing architecture. Typical noninterference models are not suitable to certificate information flow in cloud computing architecture. In this paper, we propose several information flow models for cloud architecture. One model is for transitive cloud computing architecture. The others are for intransitive cloud computing architecture. When concurrent access actions execute in the cloud architecture, we want that security domain and security domain do not affect each other, that there is no information flow between security domains. But in fact, there will be more or less indirect information flow between security domains. Our models are concerned with how much information is allowed to flow. For example, in the CIP model, the other domain can learn the sequence of actions. But in the CTA model, the other domain can't learn the information. Which security model will be used in an architecture depends on the security requirements for that architecture.

Keywords: Cloud computing security, information flow security, noninterference, noninterference models.

1 Introduction

Cloud computing provides a variety of services over the network. This open environment, while being convenient for users to access and use, also poses a potential security risk and leads to an increasing number of attacks from inside or outside the cloud. In cloud computing, there is a risk of data leakage between virtual machines or users. Whether it is direct or indirect data leakage, it can be regarded as illegitimate flow of information. Approaches such as access control the explicit flow of information, implicit flow of

[1] Nanjing Audit University, Nanjing, 211815, China.

[2] Center for Data Science, Courant Institute of Mathematical Sciences, New York University, New York, 10011-8868, USA.

[*] Corresponding Author: Congdong Lv. Email: lvcongdonglv@163.com.

Received: 28 April 2020; Accepted: 20 May 2020.

information cannot be controlled, and a non-disruptive model is needed to detect the presence of illegitimate traffic in the cloud computing infrastructure.

In cloud computing, multi-user access to resources is a typical synchronization system. The system, the user's access to resources without the established priorities. When multiple users access the same resource at the same time, it can cause an access violation. Based on this kind of access conflict, malicious users get information about other users accessing the corresponding resources. For security reasons, malicious users should not obtain such information. However, the existing non-interference model cannot characterize this abnormal information flow.

A sequence of actions of a user visiting the same resource at a time is taken as a motion vector or set of actions. In order to prevent such abnormal information flow, there should be no flow of information between the action vectors or the generating domains of the action-focused action or the flow of information should satisfy the rules of information flow stipulated by the system. In view of the existing non-interference model can only deal with the sequence of sequential actions and cannot handle the set of simultaneous actions, a variety of methods for processing simultaneous sets of actions are proposed and based on these methods, the information flow suitable for cloud computing architecture Analyzed non-interference models. Among them, the first model is suitable for analyzing information flow-secure of cloud computing architecture under transitive policy. The others are applicable to cloud computing architecture information under intransitive policy. The difference between these models lies in the different control of the system flow of information strength.

2 Related works

In cloud computing, there are works on the cloud components about the theory of information flow security, such as the formal description and analysis, verification and so on. Early, a complete functional formal proof of the operating system microkernel was performed. It provided a way to formally analyze complex systems [Chen, Wu, Shao et al. (2016)]. Then, a security tool was used to retrofit KVM, showing how their approach can support a widely-used full-featured hypervisor integrated with a commodity operating system. The implementation had a trusted computing base of only a few thousand lines of code, many orders of magnitude less than KVM. They showed that the security tool protected the confidentiality and integrity of virtual machines running unmodified guest operating systems while only incurring modest performance overhead for real application workloads [Li, Koh and Nieh (2019)]. Virtual machine migration will take a lot of time. And in the process of migration, their security will become a problem [Li, Li, Zhang et al. (2019)]. One solution made use of mainstream tools and architectures, like the Linux integrity measurement architecture, the open attestation platform and the Docker container engine, making it practical and readily available in a real-world scenario. Compared to a standard Docker deployment, their solution enabled run-time verification of container applications at the cost of a limited overhead [De and Lioy (2019)]. At the same time, a privacy-aware multi-authority cipher text policy ABE scheme with accountability was proposed, which hid the attribute information in the cipher text and allowed to trace the dishonest user identity who shared the decryption key. The efficiency analysis

demonstrated that the new scheme was efficient, and the computational overhead in the tracing algorithm was only proportional to the length of the identity [Li, Chen, Chow et al. (2018)]. The API is a bridge between cloud service providers and users. The security of cloud computing relies heavily on the security of the API. RBAC-based two-phase access control mechanism was put forward on the cloud-based API level. The access mechanism allowed only white-list users to access [Rizvi and Mitchell (2015)]. In the analysis of cloud computing environment, security is the biggest problem [Zhang, Chang, Yan et al. (2019)]. A non-interference model was proposed as the basis for analysis of the feasibility of the behavior, but did not consider the concurrency of behavior in cloud computing [Zhang, Zhang, Chen et al. (2017)].

In information flow security analysis of cloud computing, the flow of key information based on virtualized systems using DFL (Data Flow Logic) was discussed [West, Li and Missimer (2016)]. The benefits of different users in the cloud computing environment was analyzed. Chinese wall model was used in the cloud computing environment [Zeng, Koutny and Watson (2015)]. The current disk storage reuse in cloud storage was analyzed from the perspective of privacy data stealing. Technology, the issue of using illegal information flow to steal data from a target virtual machine by using methods such as hidden channel attacks in a cloud environment was studied [Srivastava and Kumar (2015)]. The advantages and disadvantages of a multi-user SaaS environment were analyzed, especially between users whether it can be properly isolated for the information flow security issues such as resource management and resource scheduling [Bzemer and Zaidman (2010)]. A hidden channel was constructed to steal other user data from disks that were not properly physically isolated by measuring file access time [Wang, Sun and Jajodia (2012)]. Security level of the cloud environment in different security domains between the isolation problems was discussed. A hypervisor-based isolated kernel was designed to achieve different levels of sensitive data isolation [Li, Yan, Chen et al. (2019]. A broadcast proxy re-encryption was proposed for data sharing [Ge, Liu, Xia et al. (2019)]. A secure data query framework for cloud computing was proposed. Cloud service was used to check queried data from fog network when fog network provided queried data to users [Gu, Wu, Yin et al. (2020)].

Information flow security is a top priority for a series of security issues in cloud computing. Because cloud computing system is too large and complex, it is difficult to formally describe and analyze it. In the discussion of isolation of the secure kernel, when one virtual machine performs an operation, the other virtual machines should be completely unaware of the fact that no information flows between the virtual machines [Zhao, Sanán and Zhang (2016)]. The internal operations of a virtual machine do not result in changes in the system state visible to other virtual machines and that no information flows between the virtual machines [Xu, Liu and Zhu (2017)].

The "perception" and "impact" of virtual machines and virtual machines fit into the semantics provided by a non-disruptive model in the field of information flow. Therefore, from the perspective of information flow, this paper analyzes the characteristics of information exchange in cloud computing environment, and formally describes and verifies the information flow in the cloud environment as a whole. The corresponding non-interference model is proposed and the non-interference model is used the flow of

information in the cloud computing infrastructure is controlled to verify the implicit flow of information in the cloud computing infrastructure [Navamani, Yue and Zhou (2018)]. The interference-free model ensures that there is no uncontrolled flow of information in the cloud computing architecture, including explicit and implicit flows. Controlling the flow of information in a cloud environment can circumvent many kinds of security threats that may lead to the disclosure of sensitive data, including the escape of virtual machines, the stealing of virtual machines, and the hidden channels in cloud computing [Di and Lombardi (2018)]. Therefore, the formal research of cloud computing environment based on information flow has strong practical significance.

The ultimate goal of introducing a non-interference model is to ensure that there is no illegitimate flow of information in the system, so how to verify that the system meets the non-interference model has always been an important topic. The usual verification method is based on the "extended theorem" [Zhao, Liu and Guo (2018)]. The method first constructs an "extension theorem" and then reduces the global constraint of the information flow security attribute to a part that involves only one-step state transition Verification of conditions. This technique establishes the relationship between system security and a single state transition command, hence the term "extension theorem". At present, this method effectively solves the verification problem of non-interference attributes on transitive systems. However, this method is no longer suitable for systems without transitive [Schoepe (2018)]. The main problem is that the method is reliable, but not complete. Reliability means that if the local conditions are satisfied, the system can be concluded that the security attributes are satisfied, Incompleteness means that if the local conditions are not met, it cannot be concluded that the system does not satisfy the security attributes. In addition to "Extended Theorem", some domestic and foreign scholars have also developed some other verification methods for non-interference attributes. The verification problem of non-intrusive model was transformed into the problem of language inclusion of automaton, which is incomplete [Georget, Jaume and Piolle (2017)]. The verification of unperturbed attributes on deterministic systems was reduced to reachability problems, which can then be verified with the aid of reachability detection techniques [Barthe, Betarte and Campo (2019)]. Then verify the cloud computing architecture to meet the interference-free model? In this paper, the "extended theorem" of non-interference model in cloud computing is given and the decision algorithm to satisfy the non-interference model is given according to "extended theorem".

3 Cloud security models

In order to facilitate the formal description of the model, the elements involved in the model are mathematically defined as shown in Tab. 1.

In cloud computing, multiple actions may occur at the same time t, called a concurrent action set, writing $a = \{a_1, \ldots, a_n\}$, where the elements in the set are repeatable.

Table 1: Cloud architecture

Element Set	Elements and their meanings	Description
U	$U = \{U_1, U_2, \ldots, U_n\}$, U_i represents a user in the cloud computing environment	Cloud user set
V	$V = \{V_1, V_2, \ldots, V_n\}$, V_i represents a server in a cloud computing environment	Cloud server set
D	$D = U \cup V$, D_i represents a security zone in a cloud computing environment	Domain set
S	$S = \{s_0, s_1, s_2, \ldots, s_n\}$, s_i represents cloud status, s_0 represents the initial status	Cloud status set
AC	$AC = \{r, w\}$, r is read, w is write	User behavior set
T	$T = \{T_1, T_2, \ldots, T_n\}$, T_i represents the tense of a behavior	Time status set
E	$E = \{E_1, E_2, \ldots, E_n\}$, E_i represents the environment of a behavior	Environmental status set
A	$A = \{AC, T, E\}$, AC, T and E constitute behavior set	Behavior set
O	$O = \{O_1, O_2, \ldots, O_n\}$, O_i represents a user view in the cloud computing environment	User view set
obs	$obs_\mu(s_i), \mu \in D, s_i \in S$, output function	Security domain in a state of view
dom	$dom(a), a \in A$, function to get the action to take place in the security domain	Occurrence of action Security domain
$step$	$step(s \cdot \alpha, a)$, $s \in S$, $a \in A, \alpha \in A^*$, $s \cdot \alpha a$ is for short, $s \cdot \varepsilon = s$, ε is null	Step by step function

→	Information flow policy	information flow between domains

Concurrent and asynchronous hybrid cloud computing architecture, both concurrent sets of actions, there are asynchronous action sequences. Traditional interference-free models can not handle the flow of information in this hybrid cloud computing architecture and need to be scaled up to create a non-disruptive model that is suitable for hybrid cloud computing architectures.

In passing the cloud computing architecture, a non-disruptive formal semantics is described based on the function cpurge. Given a policy →, the function is defined as: $A^* \times D \mapsto A^*$, $cpurge_\mu(\alpha)$ represents a subsequence of action a where $a \in \alpha$ and $dom(a) \to \mu$. In the action sequence $a \cdot \alpha$, where a is a single action, not a concurrent action set, the function $cpurge_\mu()$ is formalized as shown in Eq. (1). $cpurge_\mu(a \cdot \alpha) =$

$$\begin{cases} \varepsilon, a \cdot \alpha \text{ is null} \\ a \cdot cpurge_\mu(\alpha), \text{ if } dom(a) \to \mu \\ cpurge_\mu(\alpha), \text{if } dom(a) \nrightarrow \mu \end{cases} \quad (1)$$

when $a = \{a_1, \ldots, a_n\}$ is a concurrent action set, the formal formulation of function $cpurge_\mu()$ is as in Eq. (2).

$$cpurge_\mu(\{a_1\} \cup \{a_2, \ldots, a_n\} \cdot \alpha) =$$

$$= \begin{cases} \varepsilon, a \cdot \alpha \text{ is null} \\ \{a_1\} \cup cpurge_\mu(\{a_2, \ldots, a_n\} \cdot \alpha), \text{if } dom(a_1) \to \mu \\ cpurge_\mu(\{a_2, \ldots, a_n\} \cdot \alpha), \text{others} \end{cases} \quad (2)$$

Among them, the set operator ∪ allows the same element in the collection.

For cloud computing architecture AR, transfer policy →, action sequence $\alpha \in A^*$ and security domain $\mu \in D$, where α may contain one or more sets of concurrent actions, both of which have $obs_\mu(s_0 \circ \alpha) = obs_\mu\left(s_0 \circ cpurge_\mu(\alpha)\right)$, then the cloud computing architecture AR for policy → is CP secure. That is, the view of each security realm is affected only by the actions that affect the security realm μ. The formal description of equivalence is as follows:

Definition 3.1 For the hybrid cloud computing architecture AR, transfer policy →, action sequence $\alpha, \alpha' \in A^*$ and security domain μ. When $cpurge_\mu(\alpha) = cpurge_\mu(\alpha')$, have $obs_\mu(s_0 \circ \alpha) = obs_\mu(s_0 \circ \alpha')$, then said cloud computing architecture AR for the policy → is CP-Secure.

In a nonintransitive cloud computing architecture, for any sequence of actions that occurs arbitrarily on the security domain μ, the function $cipurge$ represents the largest sequence of actions to the action sequence that can affect μ. This definition is based on the function $csource_\mu: A^* \times D \mapsto P(D)$. *csource* represents a set of fields in an action sequence that produce an action on a flow of information in a secure domain. For the action sequence $a \cdot \alpha$, where c is a formalized expression as in Eq. (3) when a is a single action rather than a concurrent action set.

$csource_\mu(a \cdot \alpha) =$

$$= \begin{cases} \{\mu\}, a \cdot \alpha \text{ is null} \\ csource_{\mu}(\alpha) \cup \{dom(a)\}, \exists v: v \in csource_{\mu}(\alpha) \wedge dom(a) \to v \\ csource_{\mu}(\alpha), \text{ others} \end{cases} \tag{3}$$

when $a = \{a_1, \ldots, a_n\}$ is a concurrent action set, the formal formulation of function $csource$ is as in Equation 3.4.

$csource_{\mu}(\{a_1\} \cup \{a_2, \ldots, a_n\} \cdot \alpha) =$

$$= \begin{cases} \{\mu\}, a \cdot \alpha \text{ is null} \\ csource_{\mu}(\{a_2, \ldots, a_n\} \cdot \alpha) \cup \\ \{dom(a_1)\}, \exists v: v \in csource_{\mu}(\alpha) \wedge dom(a_1) \to v \\ csource_{\mu}(\{a_2, \ldots, a_n\} \cdot \alpha), \text{ others} \end{cases} \tag{4}$$

$csource_{\mu}(\alpha)$ represents a set of action-producing domains for the security domain μ with information flow in action sequence α, which contains the security of direct or indirect information flow for the security domain μ in all action sequences Eq. (3) can handle the sequence of actions, you can also use Eq. (4) processing. Therefore, the function $csource_{\mu}$ is formally formulated using Eq. (4).

The function $cipurge: A^* \times D \mapsto A^*$ indicates the sequence of actions in the action sequence that have direct or indirect information flow to the security zone. For the action sequence $a \cdot \alpha$, where a is a single action, its formal formulation is as in Eq. (5).

$cipurge_{\mu}(a \cdot \alpha) =$

$$= \begin{cases} \varepsilon, a \cdot \alpha \text{ is null} \\ a \cdot cipurge_{\mu}(\alpha), dom(a) \to \omega \in csource_{\mu}(\alpha) \\ cipurge_{\mu}(\alpha), \text{ others} \end{cases} \tag{5}$$

when $a = \{a_1, \ldots, a_n\}$ is a concurrent action set, the formal formulation of function $cipurge$ is as in Eq. (6).

$cipurge_{\mu}(\{a_1\} \cup \{a_2, \ldots, a_n\} \cdot \alpha) =$

$$= \begin{cases} \varepsilon, a \cdot \alpha \text{ is null} \\ \{a_1\} \cup cipurge_{\mu}(\{a_2, \ldots, a_n\} \cdot \alpha), \text{if } dom(a) \to \omega \in csource_{\mu}(\alpha) \\ cipurge_{\mu}(\{a_2, \ldots, a_n\} \cdot \alpha), \text{others} \end{cases} \tag{6}$$

The set operator ∪ here allows for the same element in the set. A single action is a special case of an action set. The action sequences that can be handled by Eq. (5) can also be handled using Eq. (6). The function $cipurge$ is formally formulated using Eq. (6).

Definition 3.2 For the intransitive policy →, the action sequence $\alpha, \alpha' \in A^*$ and the security domain μ, if $cipurge_{\mu}(\alpha) = cipurge_{\mu}(\alpha')$, there are $obs_{\mu}(s_0 \circ \alpha) = obs_{\mu}(s_0 \circ \alpha')$, then hybrid cloud computing architecture AR for policy → is CIP-secure.

Given a set X and A, the set $T(X, A)$ represents a minimal set containing the condition "if $x, y \in T$ and $z \in A$, $(x, y, z) \in T$." That is, $T(X, A)$ is a triplet tree, the left node and the middle node come from the intermediate calculation value, and the left node and the middle node of the leaf node are ε and the right node is the action in the set A.

Given a policy →, for any security domain $\mu \in D$, the function $cta{:}A^* \longmapsto T(\{\varepsilon\}, A)$ for the action sequence $\alpha \cdot a$, where α is a single action and a is a single action When, the concrete expression is as shown in Eq. (7).

$$cta_\mu(\alpha \cdot a) = \begin{cases} \varepsilon, a \cdot \alpha \; is \; null \\ \left(cta_\mu(\alpha), cta_{dom(a)}(\alpha), a\right), dom(a) \to \mu \\ cta_\mu(\alpha), \text{ others} \end{cases} \tag{7}$$

When $a = \{a_1, \ldots, a_n\}$ is a concurrent action set, the first problem to be solved is which actions in $\{a_1, \ldots, a_n\}$ have information flow on the security domain μ, and what are the generating domains of these actions. Function $sourceta_\mu(){:}\, A^* \times D \longmapsto P(D)$, the input parameters for the concurrent action set, the output of the security domain set. The set of output security domains is a collection of generated domains for actions that concentrate on the security domain μ for input concurrent actions. $sourceta$ is formally described as Eq. (8).

$$sourceta_\mu(\{a_1, \ldots, a_{n-1}\} \cup \{a_n\}) =$$

$$= \begin{cases} \varepsilon, a \cdot \alpha \text{ is null} \\ sourceta_\mu(\{a_1, \ldots, a_{n-1}\}) \cup \{dom(a_n)\}, \text{if } dom(a_n) \to \mu \\ sourceta_\mu(\{a_1, \ldots, a_{n-1}\}), \text{ others} \end{cases} \tag{8}$$

Function $asourceta_\mu(){:}\, A^* \times D = A^*$, the input parameters for the concurrent action set, the output for the concurrent action set. Output Concurrent Sets of Actions As a collection of inputs that concurrently concentrate on actions that have a stream of information on the security domain μ, a formal description is given in Eq. (9).

$$asourceta_\mu(\{a_1, \ldots, a_{n-1}\} \cup \{a_n\}) =$$

$$= \begin{cases} \varepsilon, a \cdot \alpha \text{ is null} \\ asourceta_\mu(\{a_1, \ldots, a_{n-1}\}) \cup \{a_n\}, if \; dom(a_n) \to \mu \\ asourceta_\mu(\{a_1, \ldots, a_{n-1}\}), \text{ others} \end{cases} \tag{9}$$

Based on function $sourceta_\mu()$ and function $asourceta_\mu()$, function $ta_\mu()$ can be described as Eq. (10).

$$cta_\mu(\alpha \cdot \{a_1, \ldots, a_n\}) =$$

$$= \begin{cases} \varepsilon, \alpha \cdot a \text{ is null} \\ \begin{pmatrix} cta_\mu(\alpha), cta_{sourceta_\mu(\{a_1,\ldots,a_n\})}(\alpha), \\ asourceta_\mu(\{a_1, \ldots, a_n\}) \end{pmatrix}, \text{if } dom(a) \to \mu \\ cta_\mu(\alpha), \text{ others} \end{cases} \tag{10}$$

In function $cta_{sourceta_\mu(\{a_1,\ldots,a_n\})}(\alpha)$, we need to extend the function cta_μ. Let X be a security domain set, cta_μ extended function cta_X described as in Eq. (11).

$$cta_X(\alpha \cdot a) =$$

$$= \begin{cases} \varepsilon, \alpha \cdot a \text{ is null} \\ \left(cta_X(\alpha), cta_{dom(a)}(\alpha), a\right), if \; \exists v{:}\, v \in X \wedge dom(a) \to v \\ cta_X(\alpha), \text{others} \end{cases} \tag{11}$$

If α is the next action in function $cta_{sourceta_\mu(\{a_1,\ldots,a_n\})}(\alpha)$, the function $sourceta$ and the function $asourceta$ need to be extended. Let X be a set of security domains, $sourceta$ extended function $sourceta_X$described in Eq. (12).

$$sourceta_X(\{a_1,\ldots,a_{n-1}\}\cup\{a_n\}) =$$
$$=\begin{cases}\varepsilon, a \text{ is null}\\ sourceta_X(\{a_1,\ldots,a_{n-1}\})\cup\{dom(a_n)\}, if\ \exists v: v\in X\wedge dom(a_n)\to v\\ sourceta_X(\{a_1,\ldots,a_{n-1}\}), others\end{cases} \quad (12)$$

$sourceta$ is extended to function $sourceta_X$, as described in Eq. (13).

$$asourceta_X(\{a_1,\ldots,a_{n-1}\}\cup\{a_n\}) =$$
$$=\begin{cases}\varepsilon, a \text{ is null}\\ asourceta_X(\{a_1,\ldots,a_{n-1}\})\cup\{a_n\}, \text{if } \exists v: v\in X\wedge dom(a_n)\to v\\ asourceta_X(\{a_1,\ldots,a_{n-1}\}), \text{others}\end{cases} \quad (13)$$

In summary, given a common processing function, the function cta as in Eq. (14).

$$cta_X(\alpha\cdot a) =$$
$$=\begin{cases}\varepsilon, \alpha\cdot a \text{ is null}\\ \left(cta_X(\alpha), cta_{source_X(a)}(\alpha), asourceta_X(a)\right), \text{if } asourceta_X(a)\neq\Lambda\\ cta_X(\alpha), \text{ others}\end{cases} \quad (14)$$

Definition 3.3 For the non-delivery policy →, the action sequence $\boldsymbol{\alpha, \alpha'\in A^*}$ and the security domain $\boldsymbol{\mu}$, there are obs when $\boldsymbol{cta_\mu(\alpha)=cta_\mu(\alpha')}$, we have $\boldsymbol{obs_\mu(s_0\circ\alpha)=obs_\mu(s_0\circ\alpha')}$, then cloud computing architecture AR is CTA-secure for policy →.

The function $msourceta$ is a set of security domains, $A^*\times D^*=P(D)$, which is a set of domains that have information flow for any security domain in the security domain set. It is described as Eq. (15).

$$msourceta_X(\{a_1,\ldots,a_{n-1}\}\cup\{a_n\}) =$$
$$=\begin{cases}\varepsilon, \alpha\cdot a \text{ is null}\\ msourceta_X(\{a_1,\ldots,a_{n-1}\})\cup\{dom(a_n)\}, \text{if } \exists v: v\in X\wedge dom(a_n)\to v\\ msourceta_X(\{a_1,\ldots,a_{n-1}\}), \text{others}\end{cases} \quad (15)$$

, which $msourceta_X(\{a\})=\{a\}$ is equal to $msourceta_X(a)=\{a\}$.

Function $masourceta$ is an action set, $A^*\times D^*=A^*$, which is a set of domains that have information flow for any security domain in the security domain set. It is described as Eq. (16).

$$masourceta_X(\{a_1,\ldots,a_{n-1}\}\cup\{a_n\}) =$$
$$=\begin{cases}\varepsilon, \alpha\cdot a \text{ is null}\\ masourceta_X(\{a_1,\ldots,a_{n-1}\})\cup\{a_n\}, \text{if } \exists v: v\in X\wedge dom(a_n)\to v\\ masourceta_X(\{a_1,\ldots,a_{n-1}\}), \text{ others}\end{cases} \quad (16)$$

Function cobs represents view set, function $cobs_X(s_0)$ is described by Eq. (17). Function $cobs_X(s_0\alpha)$ is described by Eq. (18).

$$cobs_X(s_0)=\cup_{dom(a)\in X}\{obs_{dom(a)}(s_0)\} \quad (17)$$

$$cobs_X(s_0\alpha) = \bigcup_{dom(a)\in X}\{obs_{dom(a)}(s_0\alpha)\} \tag{18}$$

Function $cview$ represents the set which is construct by view and actions. It is described by Eq. (19).

$$cview_X(\alpha \cdot a) = \\ = \begin{cases} cobs_X(s_0\alpha), \alpha \cdot a \text{ is null} \\ cview_X(\alpha) masource_X(a) \circ cobs_X(s_0\alpha), \text{if } msource_X(a) \neq \Lambda \\ cview_X(\alpha) \circ cobs_X(s_0\alpha), \text{ others} \end{cases} \tag{19}$$

Given a set X and A, the set $T(X, A)$ represents a minimal set containing the condition ""if $x, y \in T$ and $z \in A$, $(x, y, z) \in T$." That is, $T(X, A)$ is a triplet tree, the left node and the middle node come from the intermediate calculation value, and the left node and the middle node of the leaf node are ε and the right node is the action in the set A.

Given the policy $\rightarrow$, for any security domain $\mu \in D$, the function $to_\mu: A^* \mapsto T((A \cup O)^*, A)$, specifically expressed as in Eq. (20).

$$cto_X(\alpha \cdot a) = \\ = \begin{cases} obs_\mu(s_0), \alpha \cdot a \text{ is null} \\ cto_X(\alpha), if\ csource_X(a) = \Lambda \\ \left(cto_X(\alpha), cview_{csource_X(a)}(\alpha), asource(a)\right), \text{others} \end{cases} \tag{20}$$

Given a set X and A, the set $T(X, A)$ represents a minimal set containing the condition "if $x, y \in T$ and $z \in A$, $(x, y, z) \in T$." That is, $T(X, A)$ is a triplet tree, the left node and the middle node come from the intermediate calculation value, and the left node and the middle node of the leaf node are ε and the right node is the action in the set A.

Given the policy $\rightarrow$, for any security domain $\mu \in D$, the function $ito_\mu: A^* \mapsto T(O(A \cup O)^*, A)$, specifically expressed as in Eq. (21).

$$cito_X(\alpha \cdot a) = \\ = \begin{cases} obs_\mu(s_0), \alpha \cdot a \text{ is null} \\ cito_X(\alpha), \text{if } csource_X(a) = \Lambda \\ (cito_X(\alpha), cview_{msource_X(a)}(\alpha), cview_{csource_X(a)-msource_X(a)}(\alpha \cdot a), \\ , asource(a)), \text{ others} \end{cases} \tag{21}$$

Definition 3.4 For any action sequence $\boldsymbol{\alpha, \alpha' \in A^*}$, and $\boldsymbol{cto_\mu(\alpha) = cto_\mu(\alpha')}$, have $\boldsymbol{obs_\mu(s_0 \circ \alpha) = obs_\mu(s_0 \circ \alpha')}$, then Hybrid Cloud Computing Architecture AR is Policy → CTO-secure.

Definition 3.5 For any action sequence $\boldsymbol{\alpha, \alpha' \in A^*}$, and $\boldsymbol{cito_\mu(\alpha) = cito_\mu(\alpha')}$, have $\boldsymbol{obs_\mu(s_0 \circ \alpha) = obs_\mu(s_0 \circ \alpha')}$, then Hybrid Cloud Computing Architecture AR is Policy → CITO-secure.

4 Extended theorems and decision algorithms of models

Because each non-interference model is defined as a recursive definition, it is difficult to determine directly, and the recursive definition needs to be extended by using the extension

theorem. This section gives extended theorems for CP-secure, CIP-secure, and CTA-secure, as well as the decision algorithm based on the extended theorem.

4.1 Extended theorems and decision algorithms of CP-secure

Given the cloud computing architecture *AR*, $\sim^{\mu}$ represents the relationship between the two states of the cloud computing architecture *AR*, given the policy →, for any $\mu \in D$, $a \in A$, $s, t \in S$, the following three conditions :

(1) Output Consistency (OC^{CP}): $s \sim^{\mu} t \Rightarrow obs_{\mu}(s) = obs_{\mu}(t)$;

(2) Single-step Consistency (SC^{CP}): $s \sim^{\mu} t \Rightarrow s \cdot a \sim^{\mu} t \cdot a$;

(3) Local Respect (LR^{CP}): $dom(a) \nrightarrow \mu \Rightarrow s \sim^{\mu} s \cdot a$

Theorem 4.1 Given the cloud computing architecture AR and policy →, the cloud computing architecture AR is CP-secure if the cloud computing architecture meets the requirements of OC^{CP}, SC^{CP}, and LR^{CP}.

According to the extended theorem, the decision algorithm of CP-secure is given. First, find all the states in the state set *S* that satisfy the relationship of $\sim^{\mu}$. Then, determine whether the state meets the output consistency, single-step consistency and policy compliance. Algorithm is as follows:

Algorithm 4.1 CP-secure decision algorithm

Input: Cloud computing architecture and architecture information flow policy

Output: Cloud computing architecture meets or does not meet CP-secure

Begin

$for\ each\ \mu \in D$// Traverse the security domain set

{

//Initialize the set *P* and $Q[\mu]$. P represents the traversed state

// $Q[\mu]$ represents the state binary satisfying the relation $\sim^{\mu}$

$P = NULL; Q[\mu] = NULL;$

$for\ each\ s \in S$ // Traverse the security status set S

{

$P = P \cup \{s\}$;// Add state *s* to set *P*

$for\ each\ t \in S - P$ // Traverse the security status set $S - P$

$if(s \sim^{\mu} t)$ // Judge whether the state *s* and *t* satisfy the relation $\sim^{\mu}$

$Q[\mu] = Q[\mu] \cup \{(s,t)\}$; //Addd (s,t) to $Q[\mu]$

}

/****Judge Output Consistency (OC^{CP})***/

$for\ each\ (s,t) \in Q[\mu]$ //Traverse the set $Q[\mu]$

//Judge whether the two state views that satisfy the relationship are the same

$if(obs_{\mu}(s) \neq obs_{\mu}(t))$

return IN_CP_Secure; // Cloud computing architecture is not CP-secure

/**** Judge Single-step Consistency (SC^{CP})***/
$for\ each\ (s,t) \in Q[\mu]$ //Traverse the set $Q[\mu]$
$for\ each\ a \in A$ //Traverse the action set A
{
Flag = False; //Set access *Flag*
$for\ each\ (s,t) \in Q[\mu]$ //Traverse the set $Q[\mu]$
//Judge whether $(s \cdot a, t \cdot a)$ satisfy the relationship $\sim^{\mu}$
$if((s \cdot a, t \cdot a) == (s,t))$
Flag = True;// Cloud computing architecture is CP-secure
$if(Flag == False)$ //$(s \cdot a, t \cdot a)$ don't satisfy the relationship $\sim^{\mu}$
return IN_CP_Secure;// Cloud computing architecture is not CP-secure
}
/****Judge Local Respect (LR^{CP})***/
$for\ each\ s \in S$ //Traverse the set S
$for\ each\ a \in A$ //Traverse the action set A
$if(dom(a) \nrightarrow \mu)$ //Judge whether $dom(a)$ flow information to security domain μ
// If the action a does not change the state, do not judge, else if the change of state, to determine whether the two states meet the relationship $\sim^{\mu}$
$if((s \cdot a) \neq s)$
{
$Flag == False$; // Set access *Flag*
$for\ each\ (s,t) \in Q[\mu]$ //Traverse the set$Q[\mu]$
//Judge whether $(s \cdot a, s)$ satisfy the relationship $\sim^{\mu}$
$if((s \cdot a, s) == (s,t))$
Flag = True;// Cloud computing architecture is CP-secure
$if(Flag == False)$ // $(s \cdot a, s)$ don't satisfy the relationship $\sim^{\mu}$
return IN_CP_Secure ;// Cloud computing architecture is not CP-secure
}
}
return CP_Secure; // Cloud computing architecture is CP-secure
End

4.2 Extended theorems and decision algorithms of CIP-secure

Given the cloud computing architecture AR, $\sim^{\mu}$ represents the relationship between the two states of the cloud computing architecture AR, given the policy $\rightarrow$, for any $\mu \in D$, $a \in A$, $s, t \in S$, the following three conditions :

(1) Output Consistency (OC^{CIP}): $s \sim^{\mu} t \Rightarrow obs_{\mu}(s) = obs_{\mu}(t)$;

(2) Single-step Consistency (SC^{CIP}): $s \sim^{\mu} t \wedge s \sim^{dom(a)} t \Rightarrow s \cdot a \sim^{\mu} t \cdot a$;

(3) Local Respect (LR^{CIP}): $dom(a) \nrightarrow \mu \Rightarrow s \sim^{\mu} s \cdot a$

Theorem 4.1 Given the cloud computing architecture AR and policy $\rightarrow$, the cloud computing architecture AR is CIP-secure if the cloud computing architecture meets the requirements of LR^{CIP}, SC^{CIP}, and LR^{CIP}.

According to the extended theorem, the decision algorithm of CIP-secure is given. First, find all the states in the state set S that satisfy the relationship of $\sim^{\mu}$. Then, determine whether the state meets the output consistency, single-step consistency and policy compliance. Algorithm is as follows:

Algorithm 4.2 CIP-secure decision algorithm

Input: Cloud computing architecture and architecture information flow policy

Output: Cloud computing architecture meets or does not meet CIP-secure

```
Begin
for each μ ∈ D// Traverse the security domain set
{
    //Initialize the set P and Q[ μ]. P represents the traversed state
    // Q[ μ] represents the state binary satisfying the relation ∼^μ
        P = NULL; Q[ μ] = NULL;
        for each s ∈ S // Traverse the security status set S
        {
                P = P ∪ {s};// Add state s to set P
                for each t ∈ S − P // Traverse the security status set S − P
                        if(s∼^μ t) // Judge whether the state s and t satisfy the relation ∼^μ
                                Q[ μ] = Q[ μ] ∪ {(s, t)}; // Addd (s, t) to Q[ μ]
    }
    /**** Judge Output Consistency (OC^CIP)***/
    for each (s, t) ∈ Q[ μ] //Traverse the set Q[ μ]
    //Judge whether the two state views that satisfy the relationship are the same
        if(obs_μ(s) ≠ obs_μ(t))
                return IN_CIP_Secure;// Cloud computing architecture is not CIP-secure
     /**** Judge Single-step Consistency (LR^CIP)***/
    for each s ∈ S //Traverse the set S
```

$for\ each\ a \in A$ //Traverse the action set A

$if(dom(a) \nrightarrow \mu)$ //Judge whether $dom(a)$ has information flow on the security domain μ

//If the action a does not change the state, do not judge, if the change of state, to determine whether the two states meet the relationship $\sim^{\mu}$

$if((s \cdot a) \neq s)$

{

$Flag == False$; // Set access *Flag*

$for\ each\ (s,t) \in Q[\mu]$ //Traverse the set $Q[\mu]$

//Judge whether $(s \cdot a, t \cdot a)$ satisfy the relationship $\sim^{\mu}$

$if((s \cdot a, s) == (s,t))$

Flag = *True*;// Cloud computing architecture is CIP-secure

$if(Flag == False)$ /// $(s \cdot a, t \cdot a)$ don't satisfy the relationship $\sim^{\mu}$

return IN_CP_Secure ;// Cloud computing architecture is not CIP-secure

}

}

/*** Judge Local Respect (LR^{CIP})***/

$for\ each\ \mu \in D$//Traverse the set D

{

//Initialize P， P represents the traversed state

$P = NULL;\ P = P \cup \{\mu\};$

$for\ each\ v \in D - P$ // Traverse the security domain set $D - P$

{

// Find a tuple that satisfies both domains simultaneously

$for\ each\ (s,t) \in (Q(\mu) \cap Q(v))$

{

$for\ each\ a \in \mu$ // Traverse every action in μ

//Judge whether the state $(s \cdot a, t \cdot a)$ satisfy the relationship $\sim^{v}$

$if((s \cdot a, t \cdot a)\ not\ in\ Q[v])$

return IN_CIP_Secure;//Not satisfied, return not CIP-secure

$for\ each\ a \in v$ //Traverse every action in v

//Judge whether the state $(s \cdot a, t \cdot a)$ satisfy the relationship $\sim^{\mu}$

$if((s \cdot a, t \cdot a)\ not\ in\ Q[\mu])$

return IN_CIP_Secure;//Not satisfied, return not CIP-secure

}

}

}

return IN_CIP_Secure; // Cloud computing architecture is CIP-secure

End

4.3 Extended theorems and decision algorithms of CTA-secure

Given the cloud computing architecture *AR*, $\sim^{\mu}$ represents the relationship between the two states of the cloud computing architecture *AR*, given the policy →, for any $\mu \in D$, $a \in A$, $s, t \in S$, the following three conditions:

(1) Output Consistency (OC^{CTA}): $s \sim^{\mu} t \Rightarrow obs_{\mu}(s) = obs_{\mu}(t)$;

(2) Single-step Consistency (SC^{CTA}):

①$s \sim^{\mu} t \wedge s \sim^{dom(a)} t \Rightarrow s \cdot a \sim^{\mu} t \cdot a$;

②$s \sim^{\mu} t \wedge a, b$ are two concurrent action of domain $\mu \Rightarrow s \cdot ab \sim^{\mu} t \cdot ba$;

(3) Local Respect (LR^{CTA}): $dom(a) \nrightarrow \mu \Rightarrow s \sim^{\mu} s \cdot a$。

Theorem 4.3 Given the cloud computing architecture *AR* and policy →, the cloud computing architecture AR is CTA-secure if the cloud computing architecture meets the requirements of OC^{CTA}, SC^{CTA}, and LR^{CTA}.

According to the extended theorem, the decision algorithm of CTA-secure is given. First, find all the states in the state set *S* that satisfy the relationship of $\sim^{\mu}$. Then, determine whether the state meets the output consistency, single-step consistency and policy compliance. Algorithm is as follows:

Algorithm 4.3 CTA-secure decision algorithm

Input: Cloud computing architecture and architecture information flow policy

Output: Cloud computing architecture meets or does not meet CTA-secure

Begin

$for\ each\ \mu \in D$// Traverse the security domain set

{

//Initialize the set *P* and $Q[\mu]$. P represents the traversed state

// $Q[\mu]$ represents the state binary satisfying the relation $\sim^{\mu}$

$P = NULL; Q[\mu] = NULL;$

$for\ each\ s \in S$ //Traverse the security status set S

{

$P = P \cup \{s\}$;// Add state *s* to set *P*

$for\ each\ t \in S - P$ // Traverse the security status set $S - P$

$if(s \sim^{\mu} t)$ // Judge whether the state *s* and *t* satisfy the relation $\sim^{\mu}$

$Q[\mu] = Q[\mu] \cup \{(s,t)\}$; // Add (s,t) to $Q[\mu]$

}

/**** Judge Output Consistency (OC^{CIP})***/

$for\ each\ (s,t) \in Q[\mu]$ //Traverse the set $Q[\mu]$

//Judge whether the two state views that satisfy the relationship are the same

$if(obs_\mu(s) \neq obs_\mu(t))$

return IN_CTA_Secure ;// Cloud computing architecture is not CTA-secure

/**** Judge Single-step Consistency (LR^{CIP})***/

$for\ each\ s \in S$ //Traverse the security status set S

$for\ each\ a \in A$ //Traverse the action set A

$if(dom(a) \nrightarrow \mu)$ // Judge whether $dom(a)$ has information flow on the security domain μ

// If the action a does not change the state, do not judge, if the change of state, to determine whether the two states meet the relationship $\sim^\mu$

$if((s \cdot a) \neq s)$

{

$Flag == False$; // Set access *Flag*

$for\ each\ (s,t) \in Q[\mu]$ //Traverse the set $Q[\mu]$

//Judge whether $(s \cdot a, s)$ satisfy the relationship $\sim^\mu$

$if((s \cdot a, s) == (s,t))$

Flag = *True*;// Cloud computing architecture is CTA-secure

$if(Flag == False)$ // $(s \cdot a, s)$ don't satisfy the relationship $\sim^\mu$

return IN_CTA_Secure ;// Cloud computing architecture is not CTA-secure

}

}

/*** Judge Local Respect (LR^{CTA})***/

/*** $\|_\mu$ represents concurrent relations ***/

$for\ each\ \mu \in D$//Traverse the security domain set D

{

$J = NULL; K[\mu] = NULL;$

$for\ each\ a\ in\ \mu$

$J = J \cup \{a\};$

$for\ each\ b\ in\ \mu - J$

$if(a \parallel_{\mu} b)$

$K[\mu] = K[\mu] \cup \{(a, b)\}$;

}

$for\ each\ \mu \in D$//Traverse the security domain set D

{

$for\ each\ (a, b) \in K[\mu]$

$for\ each\ (s, t) \in Q(\mu)$

$if((s \cdot ab, t \cdot ba)\ not\ in\ Q[\mu])$

return IN_CTA_Secure;//Not satisfied, return not CTA-secure

// Initialize the set *P,* P represents the traversed state

$P = NULL;\ P = P \cup \{\mu\}$;

$for\ each\ v \in D - P$ // Traverse the security domain set $D - P$

{

// Find the state binary that satisfies both domains simultaneously

$for\ each\ (s, t) \in (Q(\mu) \cap Q(v))$

{

$for\ each\ a \in \mu$ // Traverse every action in μ

// Judge whether $(s \cdot a, t \cdot a)$ satisfy the relationship $\sim^{\mu}$

$if((s \cdot a, t \cdot a)\ not\ in\ Q[v])$

return IN_CTA_Secure; // Not satisfied, return not CTA-secure

$for\ each\ a \in v$ // Traverse every action in v

// Judge whether $(s \cdot a, t \cdot a)$ satisfy the relationship $\sim^{\mu}$

$if((s \cdot a, t \cdot a)\ not\ in\ Q[\mu])$

return IN_CTA_Secure;// Not satisfied, return not CTA-secure

}

}

}

return CTA_Secure; // Cloud computing architecture is CTA-secure

End

5 Conclusion

In cloud computing, there are not only the concurrent execution but also the asynchronous execution. A variety of methods exist to deal with concurrent actions and asynchronous actions. Based on these methods, the concepts of CP-secure, CIP-secure, CTA-secure CTO-secure and CITO-secure.

Cloud computing architecture is divided into two cases of transmission and non-transmission. CP is suitable for delivery. CP-secure can also be applied to non-delivery scenarios, and cloud computing architecture is the most secure (as opposed to several other security models) if cloud-based security is used in a non-delivery scenario, and the availability of cloud computing infrastructure is minimal. For non-delivery cases, considering the security requirements and availability requirements of the cloud computing architecture, the cloud computing architecture meets different security models, with the highest security being CTO-secure, CITO-secure, CTA-secure and CIP-secure from highest to lowest.

In this paper, the extended theorem and decision algorithm of non-interference model are also given. By extending the theorem and decision algorithm, we can judge whether it meets the non-interference model for a given cloud computing. If the non-interference model is not satisfied, if the non-interference model is satisfied, there is no illegitimate information flow.

Funding Statement: Natural Science Research Project of Jiangsu Province Universities and Colleges (No. 17KJD520005, Congdong Lv). It proved money for this paper.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Barthe, G.; Betarte, G.; Campo, J. D. (2019): System-level non-interference of constant-time cryptography. Part I: Model. *Journal of Automated Reasoning*, vol. 63, no. 1, pp. 1–51.

Bzemer, C. P.; Zaidman A. (2010): Multi-tenant SaaS applications: maintenance dream or nightmare? *Proceedings of the Joint ERCIM Workshop on Software Evolution and International Workshop on Principles of Software Evolution*, pp. 88-92.

Chen, H.; Wu, X; Shao, Z.; Joshua, L.; Ronghui, G. (2016): Toward compositional verification of interruptible OS kernels and device drivers. *Programming Language Design and Implementation*, vol. 51, no. 6, pp. 431-447.

De, B. M.; Lioy, A. (2019): Integrity verification of docker containers for a lightweight cloud environment. *Future Generation Computer Systems*, vol. 97, no. 3, pp. 236-246.

Di, P. R.; Lombardi, F. (2018): Virtualization technologies and cloud security: advantages, issues, and perspectives. *From Database to Cyber Security*, pp. 166-185.

Ge, C.; Liu, Z.; Xia, J.; Fang, L. (2019): Revocable identity-based broadcast proxy re-encryption for data sharing in clouds. *IEEE Transactions on Dependable and Secure Computing*.

Georget, L.; Jaume, M.; Piolle, G. (2017): Verifying the reliability of operating system-level information flow control systems in Linux. *IEEE/ACM, International FME Workshop on Formal Methods in Software Engineering*, pp. 10-16.

Gu, K.; Wu, N.; Yin, B.; Jia, W. J. (2020): Secure data query framework for cloud and fog computing. *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 332-345.

Li, G.; Yan, J.; Chen, L.; Wu, J.; Lin, Q. et al. (2019): Energy consumption optimization with a delay threshold in cloud-fog cooperation computing. *IEEE Access*, vol. 7, no. 1, pp. 159688-159697.

Li, H. X.; Li, W. J.; Zhang, S. G.; Wang, H. D.; Pan, Y. et al. (2019): Page-sharing-based virtual machine packing with multi-resource constraints to reduce network traffic in migration for clouds. *Future Generation Computer Systems*, vol. 96, no. 2, pp. 462-471.

Li, J.; Chen, X.; Chow, S. S. M.; Huang, Q.; Wong, D. S. et al. (2018): Multi-authority fine-grained access control with accountability and its application in cloud. *Journal of Network and Computer Applications*, vol. 112, no. 4, pp. 89-96.

Li, S. W.; Koh, J. S.; Nieh, J. (2019): Protecting cloud virtual machines from hypervisor and host operating system exploits. *IEEE 28th Security Symposium*, pp. 1357-1374.

Navamani, B. A.; Yue, C.; Zhou, X. (2018): Discover and Secure (DaS): An automated virtual machine security management framework. *IEEE 37th International Performance Computing and Communications Conference*, pp. 1-6.

Rizvi, S.; Mitchell, J. (2015): A semi-distributed access control management scheme for securing cloud environment. *International Conference on Cloud Computing*, pp. 501-507.

Schoepe, D. (2018): Flexible information-flow control. *Chalmers University of Technology, Department of Computer Society and Engineering*, pp. 115-126.

Srivastava, H.; Kumar, S. A. (2015): Control Framework for secure cloud computing. *Journal of Information Security*, vol. 6, no. 1, pp. 12-23.

Wang, Z.; Sun, K.; Jajodia, S. (2012): Disk storage isolation and verification in cloud. *IEEE Global Communications Conference*, pp. 771– 776.

West, R.; Li, Y.; Missimer, E. S. (2016): A virtualized separation kernel for mixed-criticality systems. *ACM Transactions on Computer Systems*, vol. 34, no. 3, pp. 201-212.

Xu, X.; Liu, G.; Zhu, J. (2017): Cloud data security and integrity protection model based on distributed virtual machine agents. *IEEE International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pp. 6-13.

Zeng, W.; Koutny, M.; Watson, P. (2015): Formal verification of secure information flow in cloud computing. *Workshop on Information Security Applications*, pp. 103-116.

Zhang, F.; Zhang, C.; Chen, W.; Hu, F.; Xu, M. (2017): Noninterference analysis of trust of behavior in cloud computing system. *Journal of Computer*, vol. 45, no. 7, pp. 1-15.

Zhang, S.; Chang, Y.; Yan, L.; Sheng, Z.; Yang, F. et al. (2019): Quantum communication networks and trust management: A survey. *Computers, Materials & Continua*, vol. 61, no. 3, pp. 1145-1174.

Zhao, W.; Liu, J.; Guo, H. (2018): Etc-iot: Edge-node-assisted transmitting for the cloud-centric internet of things. *IEEE Network*, vol. 32, no. 3, pp. 101-107.

Zhao, Y.; Sanán, D.; Zhang, F. (2016): Reasoning about information flow security of separation kernels with channel-based communication. *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 791-810.